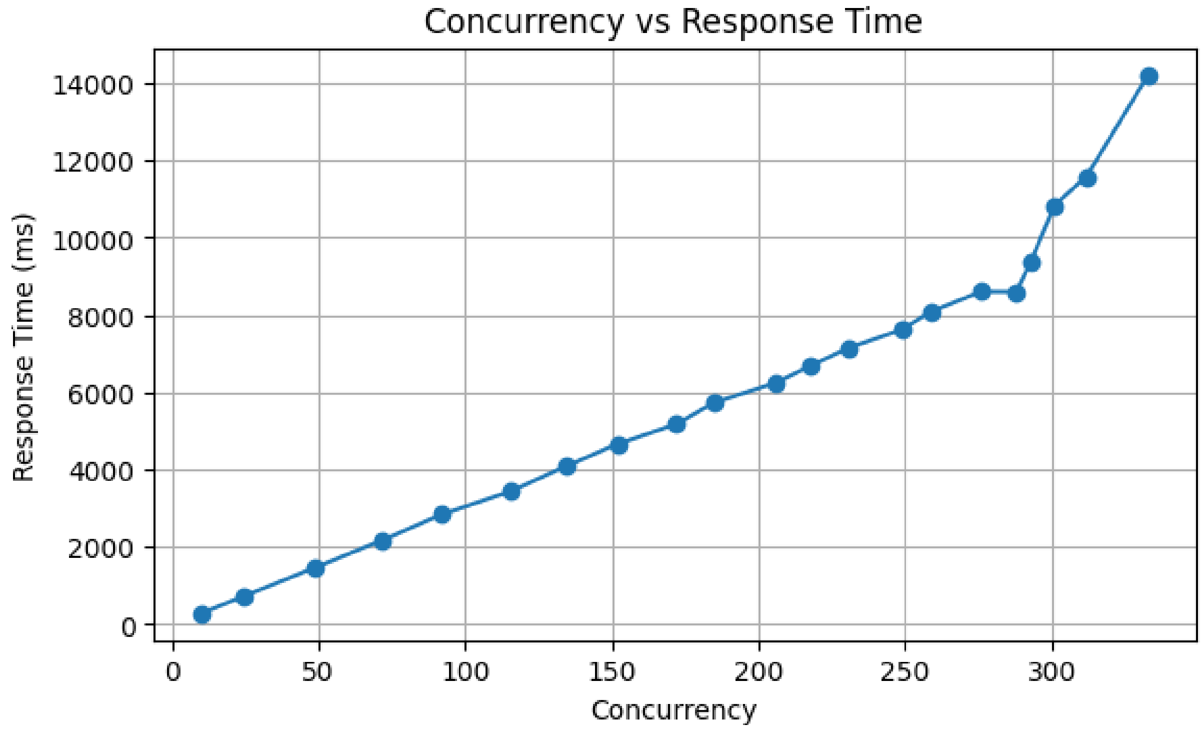
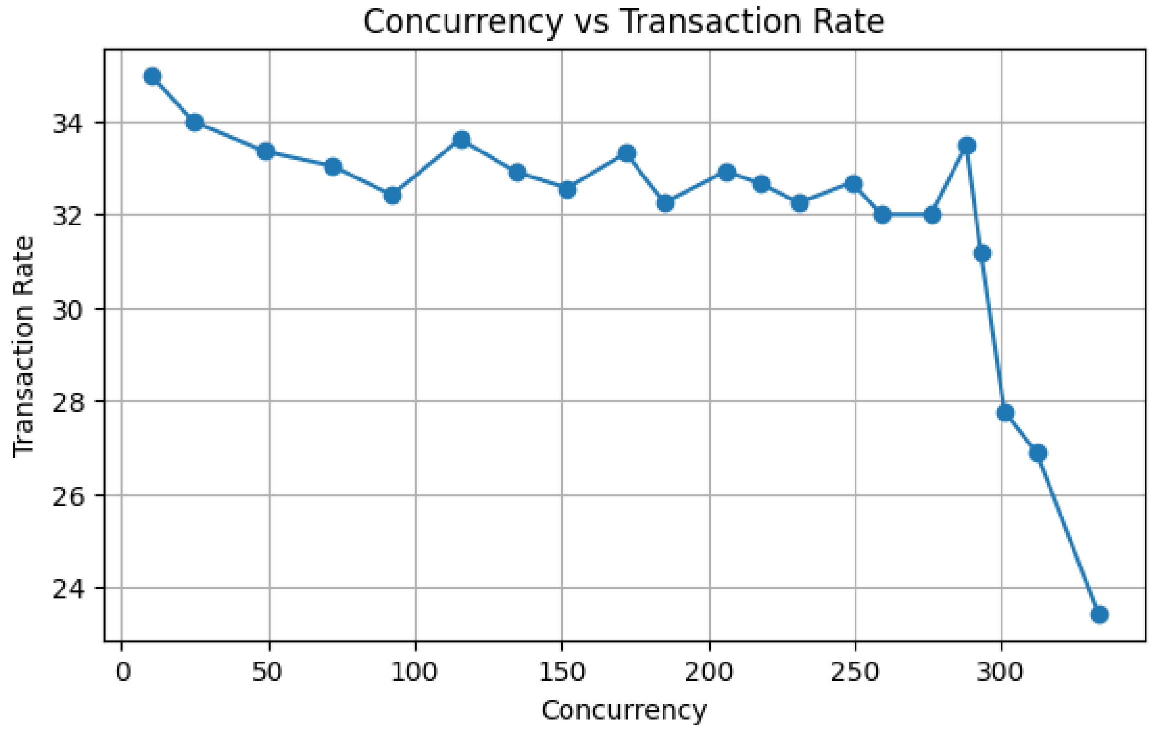
6632092821 Thanakrit Bunrueng

PaaS

Test with siege -c<vary> -d0 -t60S <hostname>

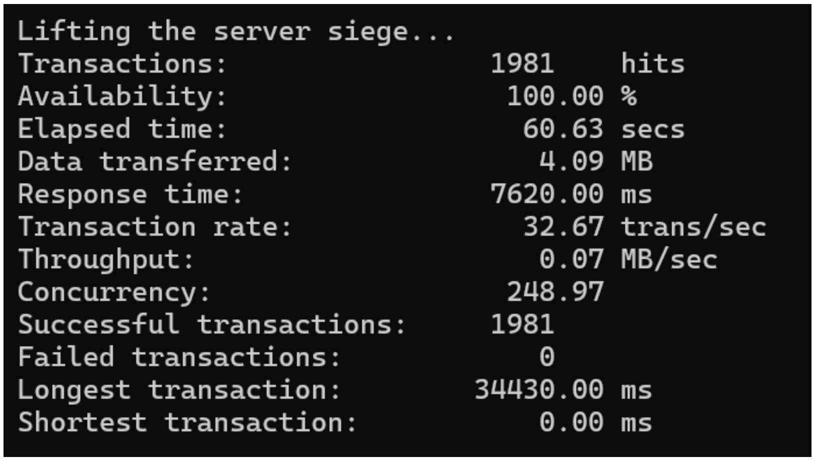




Analysis

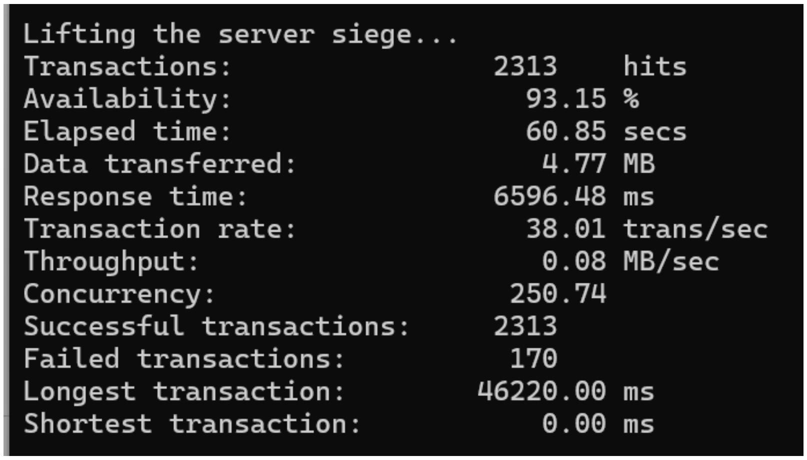
From those two graphs, the inflection point is around concurrency equal to 275 which the response time and transaction rate is advanced than usual. So, we can imply the threshold point for auto scaling with response time upper threshold > 8 seconds and lower threshold 7 seconds.

Concurrency 325



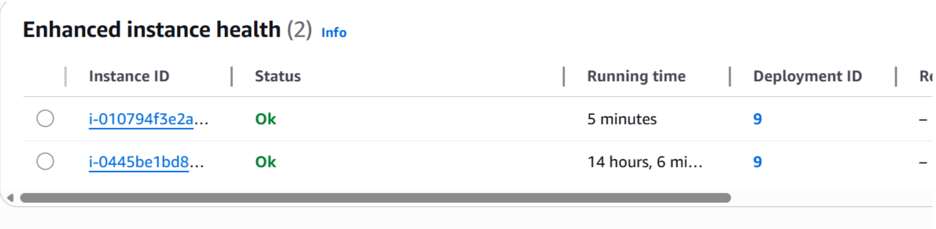
```
Lifting the server siege...
Transactions:                   1981    hits
Availability:                   100.00 %
Elapsed time:                    60.63 secs
Data transferred:                 4.09 MB
Response time:                 7620.00 ms
Transaction rate:                32.67 trans/sec
Throughput:                       0.07 MB/sec
Concurrency:                    248.97
Successful transactions:        1981
Failed transactions:               0
Longest transaction:          34430.00 ms
Shortest transaction:             0.00 ms
```

Concurrency 350 (the scaling triggered)



```
Lifting the server siege...
Transactions:                   2313    hits
Availability:                    93.15 %
Elapsed time:                    60.85 secs
Data transferred:                 4.77 MB
Response time:                 6596.48 ms
Transaction rate:                38.01 trans/sec
Throughput:                       0.08 MB/sec
Concurrency:                    250.74
Successful transactions:        2313
Failed transactions:             170
Longest transaction:          46220.00 ms
Shortest transaction:             0.00 ms
```

Its instantly triggered scaling up when the response time is greater than 8 seconds

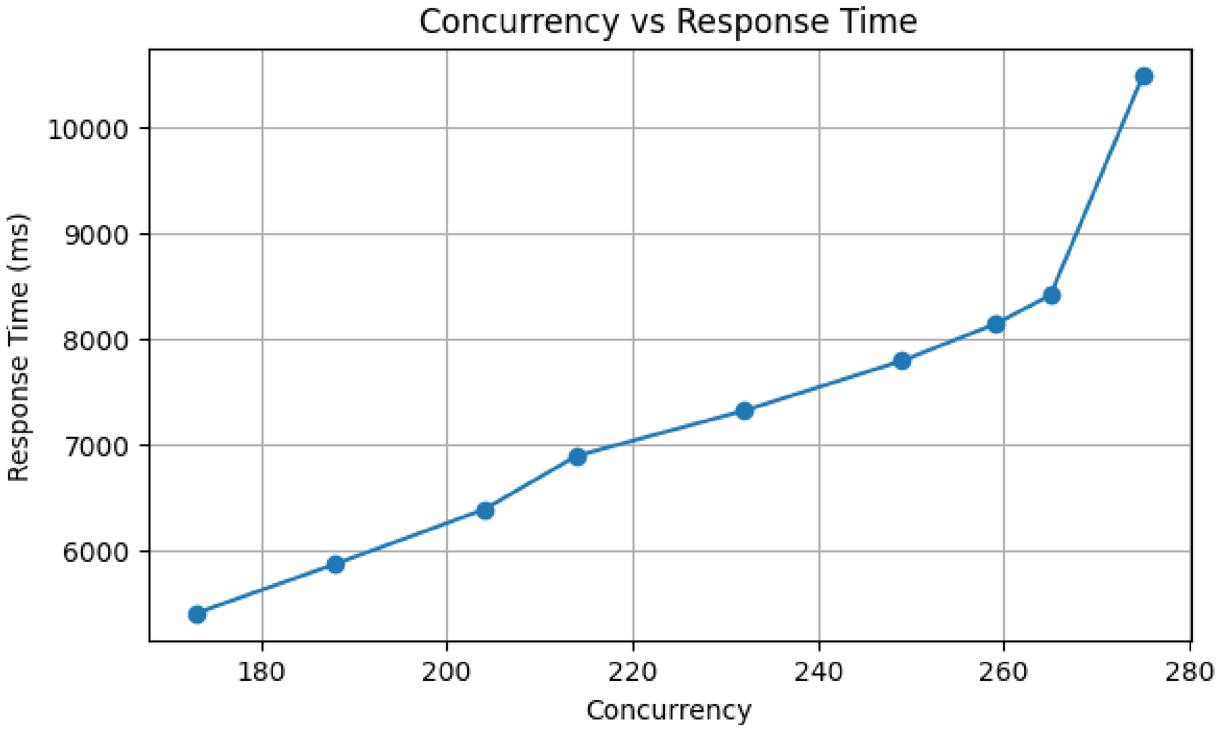


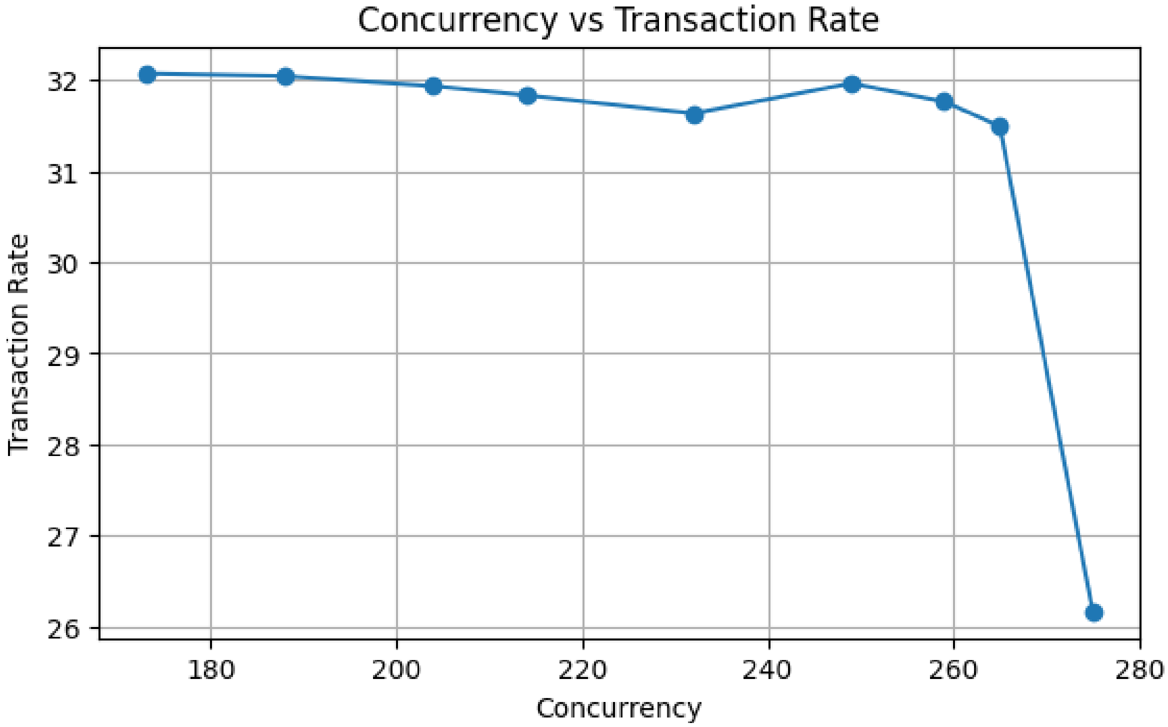
Enhanced instance health (2) Info

	Instance ID	Status	Running time	Deployment ID	Re
○	i-010794f3e2a...	Ok	5 minutes	9	–
○	i-0445be1bd8...	Ok	14 hours, 6 mi...	9	–

IaaS

Test with siege -c<vary> -d0 -t60S <hostname>



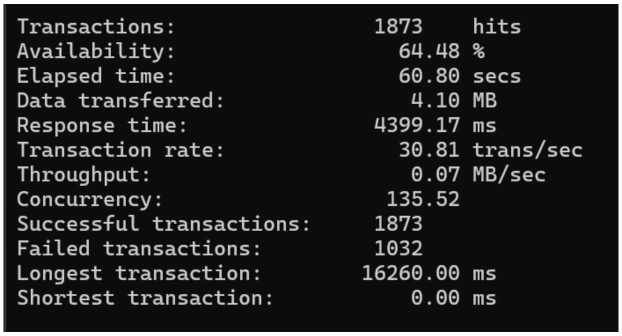


Analysis

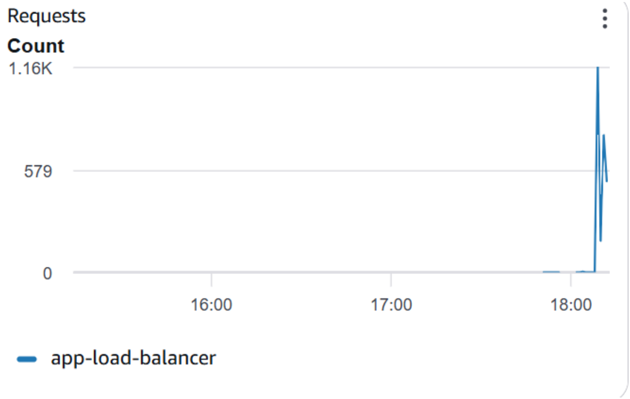
From those two graphs, the inflection point is around concurrency equal to 260. So, we can imply the threshold point for auto scaling with transaction request count with target that 1860 requests/mins.

The performance between the IaaS and PaaS according to the response time and transaction rate are not significant differences, the IaaS have slightly lower in term of offer loads. If we consider in a instance, PaaS provides t3.micro while IaaS gives t2.micro. The reason from my opinion is that PaaS required many hop between the load balancer and the elastic beanstalk stack so that why the layer is thicker than the IaaS

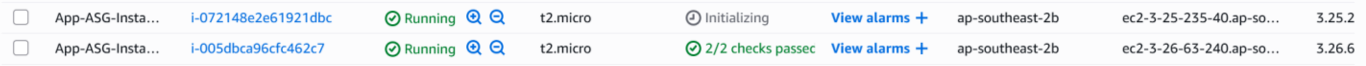
Siege -c150 -d0 -t60S <hostname>



```
Transactions:                  1873    hits
Availability:                    64.48 %
Elapsed time:                    60.80 secs
Data transferred:                 4.10 MB
Response time:                 4399.17 ms
Transaction rate:                30.81 trans/sec
Throughput:                       0.07 MB/sec
Concurrency:                   135.52
Successful transactions:       1873
Failed transactions:           1032
Longest transaction:          16260.00 ms
Shortest transaction:             0.00 ms
```



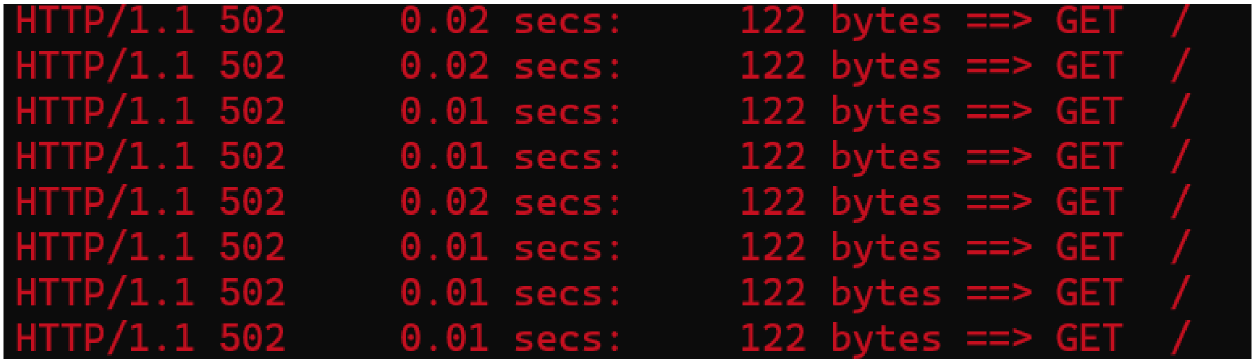
The scaling policy is consistent with request counts that I have sent.



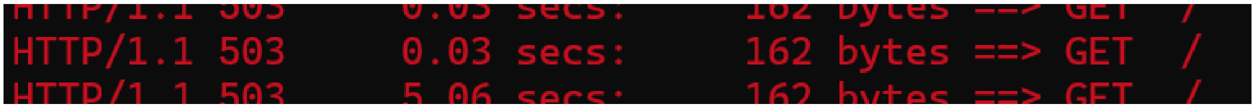
□	App-ASG-Insta...	i-072148e2e61921dbc	✓ Running	t2.micro	Initializing	View alarms +	ap-southeast-2b	ec2-3-25-235-40.ap-so...	3.25.2
□	App-ASG-Insta...	i-005dbca96cfc462c7	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	ap-southeast-2b	ec2-3-26-63-240.ap-so...	3.26.6

Fault tolerance testing

After deleting one instance

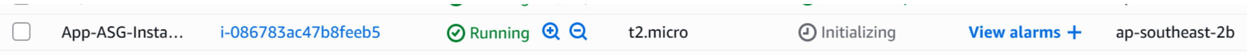


```
HTTP/1.1 502     0.02 secs:     122 bytes ==> GET  /
HTTP/1.1 502     0.02 secs:     122 bytes ==> GET  /
HTTP/1.1 502     0.01 secs:     122 bytes ==> GET  /
HTTP/1.1 502     0.01 secs:     122 bytes ==> GET  /
HTTP/1.1 502     0.02 secs:     122 bytes ==> GET  /
HTTP/1.1 502     0.01 secs:     122 bytes ==> GET  /
HTTP/1.1 502     0.01 secs:     122 bytes ==> GET  /
HTTP/1.1 502     0.01 secs:     122 bytes ==> GET  /
```



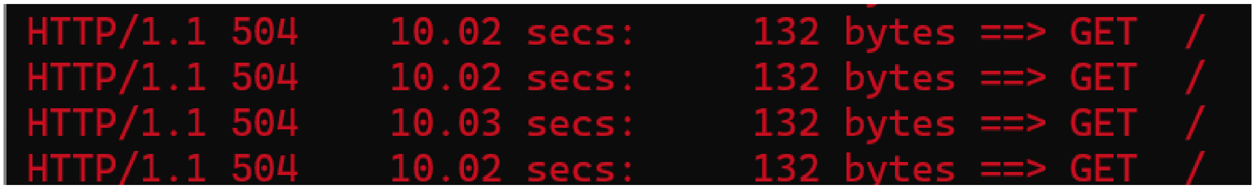
```
HTTP/1.1 503     0.03 secs:     162 bytes ==> GET  /
HTTP/1.1 503     0.03 secs:     162 bytes ==> GET  /
HTTP/1.1 503     5.06 secs:     162 bytes ==> GET  /
```

Then it creates simultaneously. But I can't access the web like normal



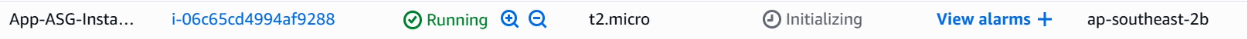
□ App-ASG-Insta... i-086783ac47b8feeb5 Running t2.micro Initializing View alarms + ap-southeast-2b

Kill all instance



```
HTTP/1.1 504    10.02 secs:     132 bytes ==> GET  /
HTTP/1.1 504    10.02 secs:     132 bytes ==> GET  /
HTTP/1.1 504    10.03 secs:     132 bytes ==> GET  /
HTTP/1.1 504    10.02 secs:     132 bytes ==> GET  /
```

It took a while to bring the one instance back



App-ASG-Insta... i-06c65cd4994af9288 Running t2.micro Initializing View alarms + ap-southeast-2b

Discussion

From configuration between those two, PaaS serves faster and easier, but it can't provide the manual config to customize like IaaS. PaaS have built everything while the IaaS have to construct manual from image to scaling group. In conclusion, the strength of IaaS is that can control and customize to optimize or tuning while the elastic beanstalk is for fast deployment and reduce operational complexity.